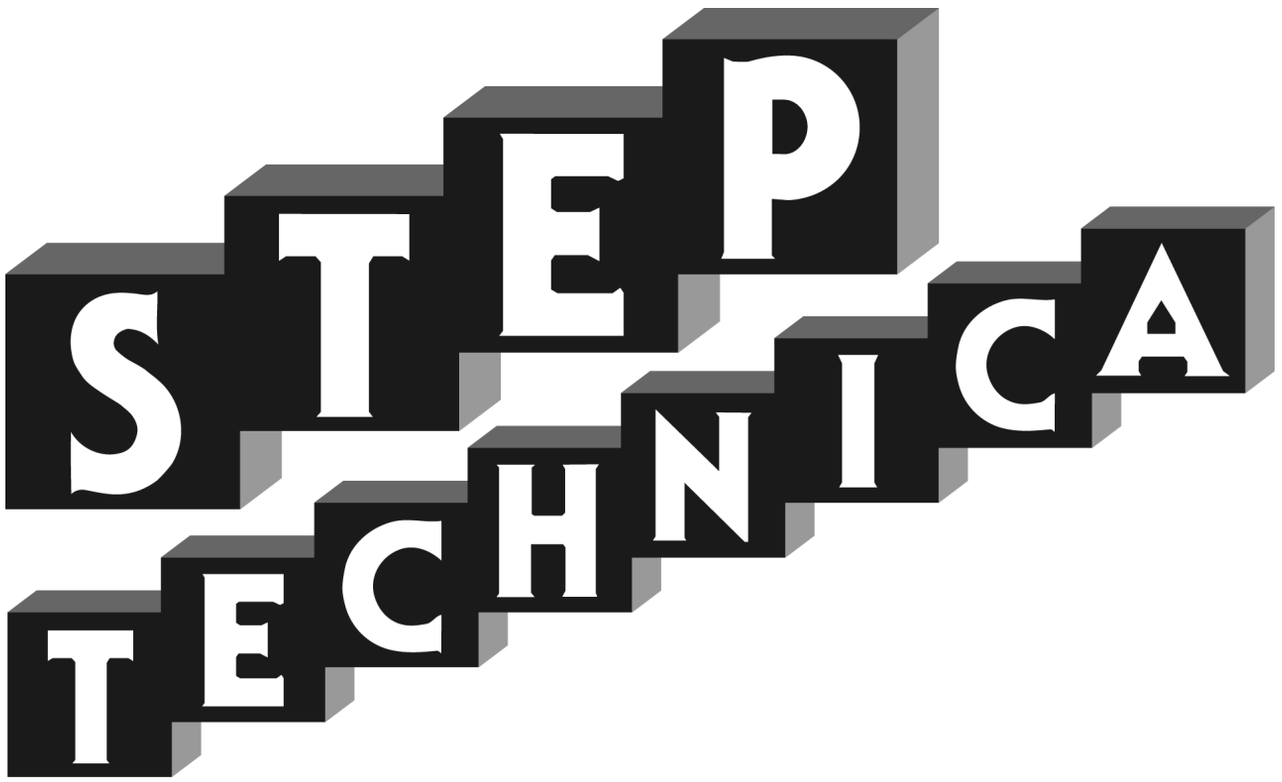
STD-HLSB36PCILP-V1.4E

STEP TECHNICA CO.,LTD.



HLS(MKY36)PCI board

HLSB-36PCI-LP

User's Manual

Note

1. The information in this document is subject to change without prior notice. Before using this product, please confirm that this is the latest version of this document.

2. Technical information in this document, such as explanations and circuit examples, are references for this product. When actually using this product, always fully evaluate the entire system according to the design purpose based on considerations of peripheral circuits and the PC board environment. We assume no responsibility for any incompatibility between this product and your system.

3. We assume no responsibility whatsoever for any losses or damages arising from the use of the information, products, and circuits in this document, or for infringement of patents and any other rights of a third party.

4. When using this product and the information and circuits in this document, we do not guarantee the right to use any property rights, intellectual property rights, and any other rights of a third party.

5. This product is not designed for use in critical applications, such as life support systems. Contact us when considering such applications.

6. No part of this document may be copied or reproduced in any form or by any means without prior written permission from StepTechnica Co., Ltd..

Preface

This manual describes the HLSB-36PCI-LP,PCI board on which the MKY36 is mounted, a kind of center IC in the Hi-speed Link System.
Be sure to read “Hi-speed Link System Introduction Guide” before understanding this manual and the HLSB-36PCI-LP.
In this manual, the Hi-speed Link System is abbreviated as “HLS.”

● Target Readers
・Those who first build an HLS
・Those who first use StepTechnica's HLSB-36PCI-LP to build an HLS

● Prerequisites
This manual assumes that you are familiar with:
・Network technology
・Semiconductor products (especially microcontrollers and memory)

● Related Manuals
・Hi-speed Link System Introduction Guide
・Hi-speed Link System Technical Guide
・Hi-speed Link System Center IC MKY36 User’s Manual

【Caution】
Some terms in this manual are different from those used on our website and in our product brochures.
The brochure uses ordinary terms to help many people in various industries understand our products.
Please understand technical information on HLS Family based on technical documents (manuals).

CONTENTS

Chapter 1　HLSB-36PCI-LP Hardware

1.1　Features of HLSB-36PCI-LP 1-1
1.2　Specifications of HLSB-36PCI-LP 1-1
1.3　Specifications of the board connectors 1-2
1.4　DIP switches of HLSB-36PCI-LP 1-3
1.5　Memory map 1-4
　1.5.1　Access to MKY36 1-4
　1.5.2　Registers unique to HLSB-36PCI-LP 1-5
1.6　Access without attached driver disc 1-5

Chapter 2　　HLSB-36PCI-LP　Software

2.1　Outline 2-1
2.2　Copyright and disclaimer 2-1
2.3　Contents of files 2-2
2.4　How to use 2-2
2.5　Limitations 2-3
　2.5.1　Multi thread 2-3
　2.5.2　Interrupt processing 2-3
2.6　API specification 2-5
　2.6.1　HlsGetVersion 2-6
　2.6.2　HlsCountDevice 2-6
　2.6.3　HlsGetLastError 2-7
　2.6.4　HlsOpenHandle 2-7
　2.6.5　HlsCloseHandle 2-7
　2.6.6　HlsReadByte , HlsReadWord 2-8
　2.6.7　HlsWriteByte , HlsWriteWord 2-8
　2.6.8　HlsGetInterrupt0Count、HlsGetInterrupt1Count 2-8
　2.6.9　HlsClearInterrupt0Count、HlsClearInterrupt1Count 2-9
　2.6.10　HlsGetInterrupt0StatusInfo、HlsGetInterrupt1StatusInfo 2-9
　2.6.11　HlsClearInterrupt0StatusInfo、HlsClearInterrupt1StatusInfo 2-10

Figures

Fig.1.1 Panel side view 1-2
Fig.1.2 Connector peripheral circuit 1-2
Fig.1.3 Settings of HLSB-36PCI-LP board 1-3

Tables

Table 1-1 Specifications of HLSB-36PCI-LP 1-1
Table 1-2 Memory map 1-4
Table 2-1 API function 2-5
Table 2-2 Internal structure of the Int0Info,Int1Info 2-9

Chapter 1 HLSB-36PCI-LP Hardware

This chapter describes the HLSB-36PCI-LP hardware.

1.1 Features of HLSB-36PCI-LP

HLSB-36PCI-LP is the HLS communication board on which Step Technica's MKY36 IC is mounted, that have expansion bus compatible with PCI. HLSB-36PCI-LP is designed for more easy operation of the MKY36 using with the attached library for Windows. Use HLSB-36PCI-LP for evaluating or learning functions of MKY36

HLSB-36PCI-LP have the 8P8C modular connector interface, and the operation can be evaluated using a straight cable for 100BASE-TX, which is CAT-5 or upper.
The operation experience with HLSB-36PCI-LP encourage operating the microcomputer system in which the MKY36 is embedded.

1.2 Specifications of HLSB-36PCI-LP

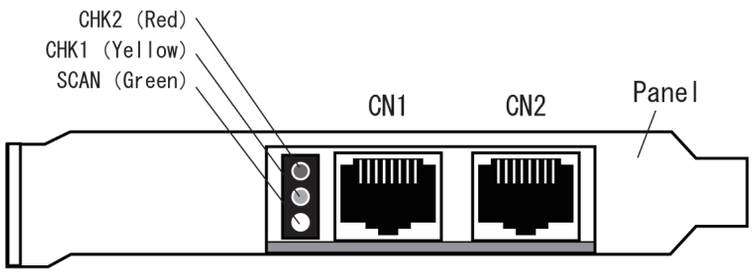
The Specifications of HLSB-36PCI-LP is given in Table 1-1.

Table 1-1 Specifications of HLSB-36PCI-LP

Type	HLSB-36PCI-LP
Type of IC mounted	MKY36 × 1
Communication method	HLS communication method
Baud rate	3M / 6M / 12M bps Full / half duplex
Compatible bus	32 bit, 33MHz expansion bus compliant with PCI Ver2.2 (Low Profile compliant)
Resources to be occupied	16KB, consecutive memory area (automatically allocated by PnP function)
Interrupt	1 line used (automatically allocated by PnP function)
Connector	RJ-45 modular connector : TM11R-5L-88 (50)
Power supply	DC +5V
Consumption current	Less than 500mA
Conditions of use	Temperature 0 – 50℃ Humidity 20 — 90% (with no condensation)
Board Size	120mm(W) x 64mm(H) *Panel side of the board is excluded.

1.3 Specifications of the board connectors

The panel side of HLSB-36PCI-LP is shown in Figure 1.



SCAN	A green LED connected to #SCANL pin of the MKY36. Lighted with starting communications with the MKY36.
CHK1	A yellow LED connected to #CHK1L pin of the MKY36. Lighted with occurrence of CHECK-1.
CHK2	A red LED connected to #CHK2L pin of the MKY36. Lighted with occurrence of CHECK-2.
CN1, CN2	HLS communication lines. A straight cable for 100BASE-TX, CAT-5 or upper is applicable to these lines. Be sure not to connect them to Ethernet board.
Panel	The panel part is touching with the PC cabinet. Connect the PC cabinet to FG with the instruction manual of the PC. To connect metal shell and No.8 pin of CN1 / CN2 to the panel, PAD1 and PAD2 on the board need to be short-circuited.

CN1 pin, CN2 pin

Pin	Function
3	RX+
4	TX-
5	TX+
6	RX-
8	Shield
1, 2, 7	Not used

Fig.1.1 Panel side view

The peripheral circuit of CN1, CN2 connectors is shown in Fig.1.2.

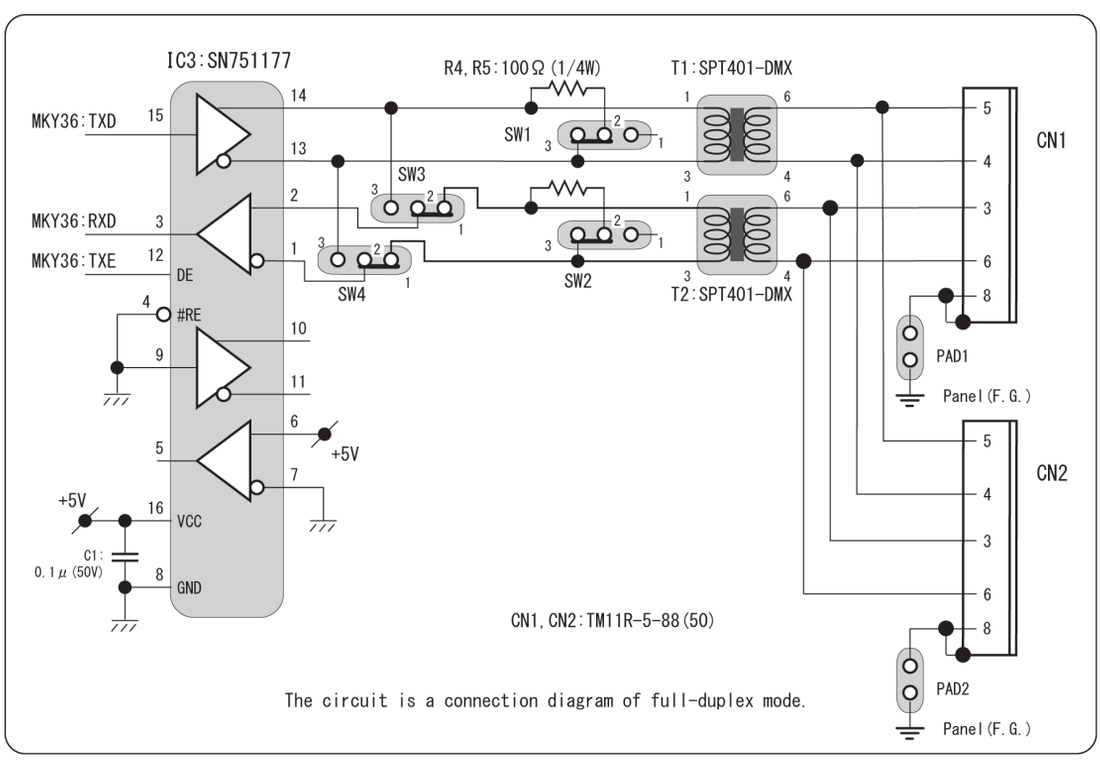


Fig.1.2 Connector peripheral circuit

1.4 DIP switches of HLSB-36PCI-LP

The dip switch settings of HLSB-36PCI-LP are shown in Fig.1.3.

To embed multiple HLSB-36PCI-LP boards in one system platform, set SW5 board ID.
This board ID is the key to identify HLSB-36PCI-LP devices by its software.

HLSB-36PCI-LP is both applicable to full-duplex and half-duplex communication mode.
To select full-duplex mode, set SW3 and SW4“Full” (1-2).
To select half-duplex mode, set SW3 and SW4“Half” (2-3).

If HLSB-36PCI-LP is at the middle position of multi-drop connection, (that is, the middle position of communication cable), set SW1 and SW2“OFF” (1-2) for termination.
If HLSB-36PCI-LP is at the terminal position of multi-drop connection, (that is, the terminus of communication cable), set SW1 and SW2 “ON” (2-3) for termination.

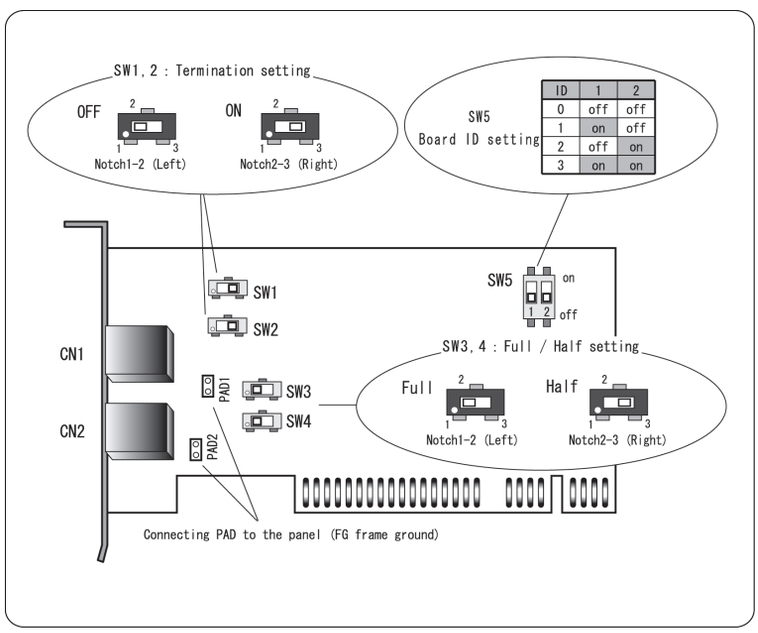


Fig.1.3 Settings of HLSB-36PCI-LP board

1.5 Memory map

Memory map for HLSB-36PCI-LP is listed in Table 1-2.
Address value in memory map is relative with a starting address of HLSB-36PCI-LP, and actual address has the value that added a starting address of the board.

Table 1-2 Memory map

Address	Function
000H ～ 07FH	Control
080H ～ 0FFH	Do area (Do)
100H ～ 17FH	Di area (Di)
180H ～ 1FFH	Counter ch1 area (C1)
200H ～ 27FH	Counter ch2 area (C2)
280H ～ 2FFH	Counter ch3 area (C3)
300H ～ 37FH	Counter ch4 area (C4)
380H ～ 3FFH	Counter ch5 area (C5)
400H ～ 47FH	Counter ch6 area (C6)
480H ～ 4FFH	SIDR: Serial Identification Register (C7)
500H ～ 57FH	DRC:Data Renewal Register
580H ～ 595H	Register
596H ～ 7FFH	Manufacturer's reserved area
C00H	Chip Reset Register
E00H	Board ID Register

1.5.1 Access to MKY36

To access to the MKY36 mounted on HLSB-36PCI-LP, write access or read access to addresses "000H — 595H" listed in Table 1-2 Memory map.
For details of each area, refer to "MKY36 User's Manual."

Do not access to addresses 596H to 7FFH (Manufacturer's reserved area.)
To do so, it may cause unstable system operation.

1.5.2 Registers unique to HLSB-36PCI-LP

C00H and E00H address registers listed in Table 1-2 Memory map are unique to HLSB-36PCI-LP. Details of C00H and E00H are given below.

Chip Reset Register Address : C00H

bit :	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	CRST
R/W :	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W

[Function] Writing "1" to CRST (Chip ResST) bit (bit 0) enables to apply reset signal to RST pin of MKY36.
Reset period to the RST pin of MKY36 is 240nS.
This register is write-only.If you read undefined data.

BoardID Register Address : E00H

bit :	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	---	---	---	---	---	---	---	---	---	---	---	---	---	---	BID1	BID0
R/W :	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

[Function] Reading BIDO,1 (BoardID 0,1) bit (bit 0,1) enables to obtain Board ID (BoardID) value (0 — 3) that is set to SW5.

1.6 Access without attached driver disc

For direct access to HLSB-36PCI-LP without Step Technica's attached driver disc, please note the following points.

Always use 32bit memory access to HLSB-36PCI-LP .
At that time, data in the lower 16 bits is valid and data in the upper 16 bits is not used.
Therefore, address value to access need to be the value which multiplied the actual address. For example, to read address 200H, read address 400H of PCI with 32bit acccess and obtain WordData in address 200H of MKY36 in the lower 16 bits out of 32 bit data that is read.
Also, to write to address 180H, write to address 300H of HLSB-36PCI-LP with 32bit access and have address 180H of MKY36 obtained the data in the lower 16bits out of 32bit that is written to PCI.
The same method is necessary to make read or write acccess to registers that are unique to HLSB-36PCI-LP.

Chapter 2 HLSB-36PCI-LP Software

This chapter describes software for using HLSB-36PCI-LP .

2.1 Outline

“DLL” folder in HLSB-36PCI-LP WDM driver disc contains the driver and DLL that are compliant to Microsoft Windows XP / 7.
At the first power-on of HLSB-36PCI-LP embedded PC, install the driver from attached disk with “Add Hardware Wizard” in Windows.
Admin rights are necessary to install the device driver. Please remember it at login.

Due to Windows specifications, all accesses to HLSB-36PCI-LP from user’s application pass through the device driver. However, since complicated process is needed for calling the device driver, API in HLSB36PCI.DLL provides more simple interface that covers its process.
At using Microsoft Windows XP / 7 as host OS, the attached software supports to use various programming tools such as Microsoft Visual Studio for easy control of HLSB-36PCI-LP.

HLSB-36PCI-LP and HLSB-36PCIEXP use its common library.
The following subchapters describe HLSB-36PCI-LP. This is applicable to HLSB-36PCIEXP.

For the latest information including the addition of compliant OS or software update, visit our website (http://www.steptechnica.com.)

2.2 Copyright and disclaimer

The copyright of all documents / program / program sources in the attached driver disc are belong to Step Technica Co., Ltd..
The individuals, companies or other parties only who acccept the cautions written below and use our HLSB-36PCI is licenced to copy or use these works of Step Technica Co., Ltd.
Step Technica Co., LTD. does not acccept not only to revise or distribute any part of these copyrighted works but also to copy and use for purposes not mentioned above without prior permission.

① Step Technica Co., Ltd. assume no responsibility for any results caused by using the attached driver disc or all softwares downloaded from our website.

② Use library in proper ways with its instructions.

③ All specifications and contents in the disc is subject to change without prior notice. Step Technica Co., Ltd. does not guarantees for any future replacements.

④ Step Technica Co., Ltd. does not support for the questions regarding OS or development environment not related to Step Technica products.

⑤ If any bugs or errors are found, contact Step Technica’s engineer team.

2.3 Contents of files

The files in “DLL” folder in the disc are given below.

【hlsb36pci.dll】

DLL body.Copy it to the system folder of Windows or the directory in which there is the user program using this DLL before use.

【hlsb36pci.lib】

Import library for Microsoft Visual C++, built with Visual Studio2010.

【hlsb36pci.h】

DLL header file. Get this included after Windows.h at use.

2.4 How to use

The files in “DLL” folder in the disc are given below.

To control HLSB-36PCI-LP device embedded in PC with hlsb36pci.dll, initializing and terminating are ncessary. The procedure is shown below.

```
//The application made for DLL version 1.xx
Int Version = HlsGetVersion();
if(Version < 0x100 || Version > 0x1FF){
   printf(“ This version of hlsb36pci.dll does not have compatibility. ＼n”);
   exit(1);
}
//1. Obtain the number of HLSB-36PCI-LP devices embedded in PC.
// (Can be omitted if it’s obvious that only one PCI board is embedded.)
Const int Count = HlsCountDevice();
If( Count < 1){
   Printf(“HLSB-36PCI-LP does not exist. ＼n”);
   exit(1);
}
//2. Obtain the handle of HLSB-36PCI-LP devices.
// (Set the parameter 0 to open if it’s obvious that only one PCI board is embedded.)
HANDLE HlsHandle[Count];
for( int I = 0;I < Count; i++){
    HlsHandle[i] = HlsOpenHandle(i);
    if( HlsHandle[i] == INVALID_HANDLE_VALUE ) exit(1);
}
//
// Access to HLSB-36PCI-LP is valid at this position.
//
//3. Close the handle which terminated the control.
for( int I = 0;I < Count;i++){
HlsCloseHandle(i);
}
```

2.5 Limitations

Describes limitations when creating an application using this API.

2.5.1 Multi thread

API in this library cannot be used at the same time from multiple threads.
Consider not to generate a collective call if the application has multithreaded structure.

2.5.2 Interrupt processing

Interrupt of MKY36,it can by setting the INT0R and INT1R, can enable or disable and confirm the interrupt status.
In the driver, manage register that holds the number of times the interrupt occurs in each INT0R and INT1R(“Interrupt generation count register”) and holds the information of the lower 8 bit of INT0R and INT1R when an interrupt occurred(“Interrupt factor register”) .
In the driver, do the processing the following using these registers when an interrupt occurs.
(Here is a description of when the interrupt INT0 has occurred.)

① Set the Interrupt factor information to "Interrupt factor register".
(Until there is a CLR instruction of the "Interrupt factor register" from the user application, the past interrupt factor remain.)
② Increment the value of the "Interrupt generation count register".
③ Clear the interrupt cause of INT0R.

API function can get information from "Interrupt generation count register" and "Interrupt factor register".
And can clear "Interrupt generation count register" and "Interrupt factor register".

(1)Function that returns the value of the interrupt generation count register.
（HlsGetInterrupt0Count、HlsGetInterrupt1Count）
Driver count the interrupts from the MKY36(to each INT0 and INT1).
This function will return this count value.
Count value will continue to be added 0-255. It will return to 0 automatically when the count value reaches 256.
(2)Function of clear "Interrupt generation count register".
（HlsClearInterrupt0Count、HlsClearInterrupt1Count）
Clear the "Interrupt generation count register".
(3)Function that returns a value of "Interrupt factor register".
（HlsGetInterrupt0StatusInfo、HlsGetInterrupt1StatusInfo）
Driver holds the interrupts factor from the MKY36(to each INT0 and INT1) .
This information is holds in "Interrupt factor register".
This function will return information of "Interrupt factor register".
(4)Function of clear "Interrupt factor register".
（HlsClearInterrupt0StatusInfo、HlsClearInterrupt1StatusInfo）
Clear the "Interrupt factor register".

The User Application can by using these functions are carried out to confirm the interrupt generation factors and count of MKY36 interrupt.

The following sample of how to check the interrupt using this API.

```
int main(void)
{
        BYTE int0_current_numOfOccurr;          // INT0 interrupt count of the current
        BYTE int0_lastTime_numOfOccurr;     // INT0 interrupt count of last time
        BYTE int0_factor;                   // INT0 interrupt factor

        /* Handle generate for HLSB-36PCI-LP */
        HlsHandle = HlsOpenHandle(0);

        /* Check the handle */
        if (HlsHandle == INVALID_HANDLE_VALUE) {
                exit(1); /* FALSE : end of program*/
        }

        /* Write FS (Final Satelite)  to SCR */
        HlsWriteWord(HlsHandle, 0x580, 0x003F);

        /* Clear Interrupt factor register */
        HlsClearInterrupt0StatusInfo(HlsHandle);
        /* Clear Interrupt generation count register */
        HlsClearInterrupt0Count(HlsHandle);

        int0_lastTime_numOfOccurr = 0;     // Interrupt generation count is 0.
        /* Set the interrupt generation factor.   CHECK-1 occur then set an interrupt INT0*/
        HlsWriteWord(HlsHandle, 0x586, 0x4000);
        while(1) {
                /* Get information of Interrupt generation count register */
                HlsGetInterrupt0Count(HlsHandle, &int0_current_numOfOccurr);

                /* Compared with the interrupt count of occurrences of the previous
                   and Interrupt occur in the case does not match */
                if (int0_lastTime_numOfOccurr != int0_current_numOfOccurr) {
                        /* Copy the current value to the previous value  */
                        int0_lastTime_numOfOccurr = int0_current_numOfOccurr;
                        /* Get information of Interrupt factor register */
                        HlsGetInterrupt0StatusInfo(HlsHandle, &int0_factor);
                        /* Interrupt generation factors whether a CHECK-1 */
                        if ((int0_factor&0x0040)==0x0040) {
                                /* --- Processing at the time of the CHECK-1 occur --- */
                                /* Clear the INT0 Interrupt factor register*/
                                HlsClearInterrupt0StatusInfo(HlsHandle);
                        }
                }
        }
        /* Close the handle that was generated */
        HlsCloseHandle(0);

        return 0;
}
```

2.6 API specification

The following API function specifications in “DLL” folder in the disc is given in this section

Table 2-1 API function

Function	Description
HlsGetVersion	Obtain the version information of library
HlsCountDevice	Obtain the number of HLSB-36PCI-LP devices
HlsGetLastError	Obtain the termination status of HLSB API function
HlsOpenHandle	Open handles to HLSB-36PCI-LP board
HlsCloseHandle	Close handles to HLSB-36PCI-LP board
HlsReadByte, HlsReadWord	Read access to HLSB-36PCI-LP board
HlsWriteByte, HlsWriteWord	Write access to HLSB-36PCI-LP board
HlsGetInterrupt0Count HlsGetInterrupt1Count	Get the interrupt count of information INT0,1 that is held inside the driver
HlsClearInterrupt0Count HlsClearInterrupt1Count	Clear the interrupt count of information INT0,1 that is held inside the driver
HlsGetInterrupt0StatusInfo HlsGetInterrupt1StatusInfo	Get the interrupt factor of information INT0,1 that is held inside the driver
HlsClearInterrupt0StatusInfo HlsClearInterrupt1StatusInfo	Clear the interrupt factor of information INT0,1 that is held inside the driver

2.6.1 HlsGetVersion

Format

UINT HLSBAPI HlsGetVersion(void);

Description

This API checks the version of library.

Parameter

None

Return value

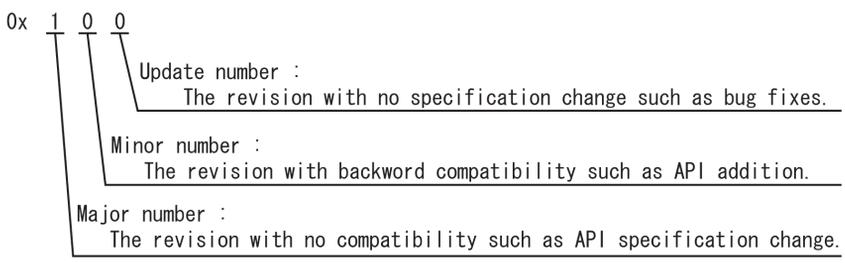
Unsigned integer value, indicates the version of library

Remark

For more secure utility, Step Technica recommends to check the compatibility with DLL via user's application using hlsb36pci.dll.. The word "secure" means that to avoid forced termination of the program by avoiding function call which is not compatible.

HlsGetVersion()API obtains the version of DLL. The hexadecimal value of version number which this API returns has meanings given in the following.

The hexadecimal number lower than second digit can be ignored because of backward compatibility, however, Step Technica recommends not to call API of hlsb36pci.dll if the value upper than third digit has been changed. Compatibility needs to be checked prior to the initialization.


0x 1 0 0
Update number :
The revision with no specification change such as bug fixes.
Minor number :
The revision with backword compatibility such as API addition.
Major number :
The revision with no compatibility such as API specification change.

2.6.2 HlsCountDevice

Format

UINT HLSBAPI HlsCountDevice(void);

Description

This API checks the number of existing HLSB-36PCI-LP device.

It's not necessary to call the function if the number of HLSB-36PCI-LP board is obvious. HlsOpenHandle() must fail if this function does not obtain the return value which is 1 or upper.

Parameter

None

Return value

-1	Over 10 boards exist
0	No boards exist
1 ～ 9	1 to 9 boards exist

2.6.3 HlsGetLastError

Format

UINT HLSBAPI HlsGetLastError(void)

Description

This API checks the termination status of API function which the process called last time.

Parameter

None

Return value

The definition in hlsb36pci.h is the following.

Non-numeric literal	Value
HLS_SUCCESS	0/* Terminated normally. */
HLS_ERR_DEVICENOTEXIST	1/* Device does not exist. */
HLS_ERR_ALREADYOPENED	2/* Handle is already opened. */
HLS_ERR_CLOSED	3/* HlsOpenHandle() has never been called*/
HLS_ERR_INVALIDPARAM	4/* Called with invalid parameter. */
HLS_ERR_NORESOUCE	5/* No resource to execute the process. */
HLS_ERR_FAILED	6/* The process failed due to unknown reason. */
HLS_NOTCALLYET	99 /*API has never been called yet. * /

2.6.4 HlsOpenHandle

Format

HANDLE HLSBAPI HlsOpenHandle(int Instance);

Description

This API returns the handle to HLSB-36PCI-LP board of which has the specified instance. It is possible to obtain multiple handles from the same instance. There is no failure due to HLS_ERR_ALREADYOPENED in current version. All API described from this section need to be called using the handle that obtained with HlsOpenHandle(). Also, close the handle by calling HlsCloseHandle() after all processes are terminated.

Parameter

Serial number of the board whose base point is Instance “0.”

The board number to obtain each handles of embedded boards. If only one board is embedded, the handle should be opened with the parameter “0.”

Return value

Succeeded : HANDLE is returned.

Failed : INVALID_HANDLE_VALUE is returned. Check for the details of failure with HlsGetLastError().

Non-numeric literal, INVALID_HANDLE_VALUE is available having windows.h included.

2.6.5 HlsCloseHandle

Format

BOOL HLSBAPI HlsCloseHandle(HANDLE HLSBHandle);

Description

This API closes the handle which is obtained with HlsOpenHandle().

Parameter

HLSBHandle Handle to be closed

Return value

Succeeded: TRUE is returned.

Failed: FALSE is returned.

2.6.6 HlsReadByte , HlsReadWord

Format

BOOL HLSBAPI HlsReadByte(HANDLE HLSBHandle, ULONG Adr, BYTE* Dat);
BOOL HLSBAPI HlsReadWord(HANDLE HLSBHandle, ULONG Adr, WORD* Dat);

Description

This API obtains BYTE / WORD data from the specified address of HLSB-36PCI-LP.

Parameter

HLSBHandle The handle to targeted HLSB-36PCI-LP.
Adr An offset address of the starting address.It must be a multiple of 2 via WORD access.
Dat A pointer to BYTE / WORD area where the obtained value is stored.

Return value

Succeeded: TRUE is returned.
Failed: FALSE is returned.

2.6.7 HlsWriteByte , HlsWriteWord

Format

BOOL HLSBAPI HlsWriteByte(HANDLE HLSBHandle, ULONG Adr, BYTE Dat);
BOOL HLSBAPI HlsWriteWord(HANDLE HLSBHandle, ULONG Adr, WORD Dat);

Description

This API writes BYTE / WORD data to the specified address of HLSB-36PCI-LP.

Parameter

HLSBHandle The handle to the targeted HLSB-36PCI-LP.
Adr An offset address of the starting address.It must be a multiple of 2 via WORD access.
Dat BYTE / WORD data where the obtained value is stored.

Return value

Succeeded: TRUE is returned.
Failed: FALSE is returned.

2.6.8 HlsGetInterrupt0Count、HlsGetInterrupt1Count

Format

BOOL HLSBAPI HlsGetInterrupt0Count(HANDLE HLSBHandle, BYTE *Int0Count);
BOOL HLSBAPI HlsGetInterrupt1Count(HANDLE HLSBHandle, BYTE *Int1Count);

Description

Get the interrupt count of information INT0,1 that is held inside the driver Parameter.
Count value will continue to be added 0-255.
It will return to 0 automatically when the count value reaches 256.

Parameter

HLSBHandle The handle to the targeted HLSB-36PCI-LP.
Int0Count、Int1Count Pointer of the area in which to store the INT0,1 interrupt count of occurrences.

Return value

Succeeded: TRUE is returned.
Failed: FALSE is returned.

2.6.9 HlsClearInterrupt0Count、HlsClearInterrupt1Count

Format

BOOL HLSBAPI HlsGetInterrupt0Count(HANDLE HLSBHandle, BYTE *Int0Count);
BOOL HLSBAPI HlsGetInterrupt1Count(HANDLE HLSBHandle, BYTE *Int1Count);

Description

Clear the interrupt count of information INT0,1 that is held inside the driver.

Parameter

HLSBHandle The handle to the targeted HLSB-36PCI-LP.

Return value

Succeeded: TRUE is returned.
Failed: FALSE is returned.

2.6.10 HlsGetInterrupt0StatusInfo、HlsGetInterrupt1StatusInfo

Format

BOOL HLSBAPI HlsGetInterrupt0StatusInfo(HANDLE HLSBHandle, BYTE *Int0Info);
BOOL HLSBAPI HlsGetInterrupt1StatusInfo(HANDLE HLSBHandle, BYTE *Int1Info);

Description

Get the interrupt factor of information INT0,1 that is held inside the driver.

Parameter

HLSBHandle The handle to the targeted HLSB-36PCI-LP.
Int0Info、Int1Info Pointer of the area in which to store the INT0,1 interrupt factor of information.

Return value

Succeeded: TRUE is returned.
Failed: FALSE is returned.

Note

Noted in Table 2-2 the configuration of the parameters that are set to Int10nfo and Int1Info.
When an interrupt occurs, a "1" bit corresponding to the occurrence factor.

Table 2-2 Internal structure of the Int0Info,Int1Info

Bit	Interrupt factor
7	Interrupt trigger due to scan stop
6	Interrupt trigger due to occurrence of CHECK-2
5	Interrupt trigger due to occurrence of CHECK-1
4	Interrupt trigger at issuing of new DREQ from the satellite IC
3	Interrupt trigger when a scan cycle is completed
2	Interrupt trigger by the DR2 function
1	Interrupt trigger by the DR1 function
0	Interrupt trigger by the DR0 function

2.6.11 HlsClearInterrupt0StatusInfo、HlsClearInterrupt1StatusInfo

Format

BOOL HLSBAPI HlsClearInterrupt0StatusInfo(HANDLE HLSBHandle);

BOOL HLSBAPI HlsClearInterrupt1StatusInfo(HANDLE HLSBHandle);

Description

Clear the interrupt factor of information INT0,1 that is held inside the driver

Parameter

HLSBHandle The handle to the targeted HLSB-36PCI-LP.

Return value

Succeeded: TRUE is returned.

Failed: FALSE is returned.

■ Developed and manufactured by

Step Technica Co., Ltd

757-3, Shimofujisawa, Iruma, Saitama

TEL: 04-2964-8804

FAX: 04-2964-7653

http://www.steptechnica.com/

info@steptechnica.com

HLS (MKY36) PCI board
HLSB-36PCI-LP
User's Manual

Document No：STD-HLSB36PCILP-V1.4E

Issued: October 2014